IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Application of: Zachary Merlynn Loafman	:
	: Before the Examiner: Duc T. Doan
Serial No: 10/660,070	:
Filed: 09/11/2003	: Group Art Unit: 2188
Title: SYSTEM AND METHOD OF SQUEEZING MEMORY SLABS EMPTY	: Confirmation No.: 6184

APPELLANTS' BRIEF UNDER 37 C.F.R. 1.192

Assistant Commissioner of Patents
Washington, D. C. 20231

Sir:

This is an appeal to a final rejection dated August 23, 2006 of Claims 1 – 5, 8 – 12, and 15 - 19 in the Application. This brief is submitted pursuant to a Notice of Appeal filed on November 30, 2006 in accordance with 37 C.F.R. 1.192.

BRIEF FOR APPLICANTS - APPELLANTS

(i)

Real Party in Interest

The real party in interest is International Business Machines Corporation (IBM), the assignee.

(ii)

Related Appeals and Interferences

There are no other appeals or interferences known to appellants, appellants' representative or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(iii)

Status of Claims

Claims 1 – 5, 8 – 12, and 15 - 19 are finally rejected and Claims 6, 7, 13, 14, 20 and 21 were objected to for being dependent on a rejected claim but were indicated as being allowable if rewritten in independent form. The rejected claims are being appealed.

(iv)

Status of Amendment

An "Amendment After Final" was not filed.

(v)

Summary of Claimed Subject Matter

The invention, as claimed in Claim 1, provides a method of squeezing slabs empty wherein a slab is a block of allocated memory space. The method comprises the steps of determining whether a slab is to be squeezed empty; and precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab (page 9, lines 16 – 24, page 10, lines 16 - 27 as well as item 325 of Figs. 3a and 3b).

The invention, as claimed in Claim 8, provides a computer program product for squeezing slabs empty wherein a slab is a block of allocated memory space. The computer program product comprises code means for determining whether a slab is to be squeezed empty; and code means for precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab (page 9, lines 16 – 24, page 10, lines 16 - 27 as well as item 325 of Figs. 3a and 3b). The code means are the steps outlined on page 11, line 7 to page 13, line 4 describing Figs. 4, 5 and 6.

The invention, as claimed in Claim 15, provides a system for squeezing slabs empty wherein a slab is a block of allocated memory space. The system comprises at least one storage system (i.e., main memory 704, memory 724, disk 726, tape 728 or DVD/CD 730) for storing code data and at least one processor (processor 702) for processing the code data to determine whether a slab is to be squeezed empty; and to preclude, if the slab is to be squeezed empty, data from being placed in any unused space of the slab (page 9, lines 16 – 24, page 10, lines 16 - 27 as well as item 325 of Figs. 3a and 3b).

(vi)

Grounds of Rejection to be Reviewed on Appeal

Whether it was proper to reject Claims 1 – 4, 8 – 11, and 15 - 18 under 35 USC §103(a) as being unpatentable over Bonwick in view of Printezis et al.

Whether it was proper to reject Claims 5, 12 and 19 under 35 USC §103(a) as being unpatentable over Bonwick in view of Printezis et al. and further in view of Benayon et al.

(vii)

Arguments

Whether it was proper to reject Claims 1 – 4, 8 – 11, and 15 - 18 under 35 USC §103(a) as being unpatentable over Bonwick in view of Printezis et al.

Claims 1, 8 and 15

Bonwick discloses a slab memory allocator. According to the teachings of Bonwick, allocation and de-allocation of objects are among the most common operations in the kernel. Thus, a fast kernel memory allocator becomes essential to enhance performance. Consequently, Bonwick devises a method of allocating memory space in slabs (a slab is, for example, a block of 16 contiguous pages) instead of allocating one page of memory at a time. As in the case of pages, slabs cannot be deallocated until empty.

However, Bonwich does not teach, show or so much as suggest ***a method of squeezing a slab empty***. Since Bonwick does not teach a method of squeezing a slab empty, he cannot have taught the step of ***determining whether a slab is to be squeezed empty*** as claimed by the Examiner.

In the "Response to Arguments" section of the Final Office Action, the Examiner remarked that "forcing a slab empty can be understood as the system dumping the data out of the memory immediately" and that no support can be found in the disclosure for such forcing action in fact the specification merely teaches a de-allocation method in which the data in memory are forced out instead the system waits until all data in a slab is not longer being used etc.

Applicants would like to point to page 9, lines 16 - 24 of the Specification where an example of "squeezing a memory slab empty" is disclosed as (1) precluding data from being written in any empty page of the slab (by disclaiming those empty pages); and (2) when a previously-used page of a slab that is being squeezed empty becomes available, data is precluded from being stored therein (by disclaiming the newly-available page). Therefore, a slab that is being squeezed empty can only become smaller and smaller until its size becomes zero; hence, the term squeezing the slab empty.

Applicants reiterate that Bonwick does not teach the step of ***determining whether a slab is to be squeezed empty***.

Further, the Examiner admitted that Bonwick does not teach the step of ***precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab***. However, the Examiner stated that Printezis et al. do teach this step. Applicants disagree.

Printezis et al. purport to teach a mostly concurrent compaction in a garbage collection system. According to the teachings of Printezis et al., the garbage collector and application programs may execute concurrently. When the garbage collector is run, it first identifies a section of memory that it wants to reclaim by selecting objects in that section of memory. The garbage collector then reserves another section of memory into which those objects are to be copied. After doing so, the garbage collector runs a "write barrier" program concurrently with identifying pointers that point to objects in the section of memory that is to be reclaimed.

The role of the "write barrier" is to mark as dirty, pointers that have been modified and that point to particular locations in the section of memory to be reclaimed in cases where the pointer modifications occur after those particular locations were scrutinized for pointers (see col. 8, lines 16 – 26). This is needed because the application programs continue to execute while the garbage collector is in use and therefore can continue to store data in the section to be reclaimed (see col. 8, lines 33 – 47). Hence, the "write barrier" enables the garbage collector to be aware of any such pointer modifications.

After concurrently executing the "write barrier" and identify the pointers, the execution of the application program is suspended in order to identify any modified pointers. Once that is done, the modified pointers may again be modified to point to a location in the reserved section of memory where the objects they point to are to be copied. Then, the objects are copied into the reserved section. The section of memory where the objects are copied from can now be added to a free list for reallocation and thus be reclaimed.

Thus, Printezis et al. do not teach, show or suggest ***precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab*** as claimed by the Examiner. Indeed, if Printezis et al. did teach the step of precluding data from being written in unused locations of a slab, they would not have defined the term "dirty bits" as indicating "modified pointers" to data in the section of memory to be disclaimed (see col. 6, lines 39 – 49). That is, there would not have been any "modified pointers" pointing to objects in the section to be disclaimed since no data would be written in that section of memory.

It is a well settled law that in considering a Section §103 rejection, the subject matter of the claim "as a whole" must be considered and analyzed. In the analysis, it is necessary that the scope and contents of the prior art and differences between the art and the claimed invention be determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966).

Since neither Bonwick nor Printezis et al. teach what the Examiner asserted that they teach, Applicants submit that the claims are allowable over the references.

Claims 2, 9 and 16

Claims 2, 9 and 16 include the limitations "wherein data is precluded from being placed in any space in the slab that becomes unused anytime thereafter."

The Examiner stated that those claims are rejected based on the same rationale as the rejection of Claims 1, 8 and 15. Applicants are not sure whether or not by this statement the Examiner is saying that Bonwick or Printezis et al. or both Bonwick and Printezis et al. teach the claimed limitations.

Nonetheless, Applicants submit that neither Bonwick nor Printezis et al. teach such limitations. As alluded to above, Bonwick teaches slab allocations and de-allocations but not a method of squeezing slabs empty. Therefore, Bonwick does not teach "precluding data from being placed in any space that becomes unused anytime thereafter in a slab that is being squeezed empty."

Printezis et al., on the other hand, does not teach the step of precluding data from being written in empty spaces of a slab; therefore, they do not teach the step of "precluding data from being placed in any space that becomes unused anytime thereafter in a slab that is being squeezed empty."

Hence, Applicants submit that these claims are also patentable.

Claims 4, 11 and 18

Claims 4, 11 and 18 include the limitations of "wherein precluding data from being placed in an unused space of the slab includes disclaiming the unused space."

As support for the rejection, the Examiner stated that in Paragraph 3.2 Bonwick describes reclaiming a slab of memory, thus including any left-over bytes that are not assigned to objects.

However, Applicants fail to see how that statement renders the claims unpatentable. The claimed limitations specifically state that to preclude data from being placed in the slab, unused space in the slab is disclaimed (i.e., the slab is made smaller in size). But based on the limitations of Claim 1, on which the instant claim depends, as well as the wording of the claim, the slab itself is not being reclaimed (note that a slab can only be reclaimed when empty). Therefore, whether or not left-over bytes that are not assigned to objects are reclaimed while the slab is reclaimed is at best irrelevant and at worse not an accurate statement since the slab has to be empty when it is reclaimed.

Applicants submit that these claims are also patentable over the applied references.

Whether it was proper to reject Claims 5, 12 and 19 under 35 USC §103(a) as being unpatentable over Bonwick in view of Printezis et al. and further in view of Benayon et al.

Claims 5, 12, and 19 recite the limitations of "wherein a collection of slabs is a pile, the pile having a maximum amount of allowable memory space that can be allocated thereto."

The Examiner admitted that neither Bonwick nor Printezis et al. teach these limitations. However, the Examiner stated that Benayon et al. describe pools of memory in which each pool is provided for a specific object memory size and a specific page list for every rounded object size. Applicants fail to understand how the statement of the Examiner describing Benayon et al. renders the claimed invention unpatentable.

Nonetheless, Benayon et al. does not teach the claimed invention. Benayon et al. teach a method for heap management of fixed sized objects using pages. Accordingly, Benayon et al. teach that fixed size data objects are allocated and de-allocated from a page list comprising a pool of memory pages and each page includes a reserved area for storing object information in common to all the objects in that page. A pool of pages, i.e. page list, is provided for each specific object size. A recycle page list is also provided for recycling pages in which all the objects have been returned.

However, nowhere in their disclosure do Benayon et al. disclose that a collection of slabs is a pile and that a pile has a maximum amount of allowable memory space that can be allocated thereto.

Therefore, Applicants submit these claims are also allowable.

Based on the foregoing, Applicants request passage to issue of all the claims in the Application.

Respectfully Submitted

By: [illegible]
Volel Emile
Attorney for Applicants
Registration No. 39,969
(512) 306-7969

(viii)

Claims Appendix

1. (Original) A method of squeezing slabs empty, a slab being a block of allocated memory space, the method comprising the steps of:

   determining whether a slab is to be squeezed empty; and

   precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab.

2. (Original) The method of Claim 1 wherein data is precluded from being placed in any space in the slab that becomes unused anytime thereafter.

3. (Original) The method of Claim 2 wherein the slab is de-allocated when the slab becomes empty.

4. (Original) The method of Claim 3 wherein precluding data from being placed in an unused space of the slab includes disclaiming the unused space.

5. (Original) The method of Claim 4 wherein a collection of slabs is a pile, the pile having a maximum amount of allowable memory space that can be allocated thereto.

6. (Original) The method of Claim 5 wherein if an application reduces the maximum amount of allowable memory space of a pile and the current amount of memory space exceeds the reduced maximum, at least one of the slabs in the pile is targeted to be squeezed empty.

7. (Original) The method of Claim 6 wherein if the program requests advice as to which area of the allocated slab to be de-allocated, the advice is returned to the program.

8. (Original) A computer program product on a computer readable medium for squeezing slabs empty, a slab being a block of allocated memory space, the computer program product comprising:

code means for determining whether a slab is to be squeezed empty; and

code means for precluding, if the slab is to be squeezed empty, data from being placed in any unused space of the slab.

9. (Original) The computer program product of Claim 8 wherein data is precluded from being placed in any space in the slab that becomes unused anytime thereafter.

10. (Original) The computer program product of Claim 9 wherein the slab is de-allocated when the slab becomes empty.

11. (Original) The computer program product of Claim 10 wherein precluding data from being placed in an unused space of the slab includes disclaiming the unused space.

12. (Original) The computer program product of Claim 11 wherein a collection of slabs is a pile, the pile having a maximum amount of allowable memory space that can be allocated thereto.

13. (Original) The computer program product of Claim 12 wherein if an application reduces the maximum amount of allowable memory space of a

pile and the current amount of memory space exceeds the reduced maximum, at least one of the slabs in the pile is targeted to be squeezed empty.

14. (Original) The computer program product of Claim 13 wherein if the program requests advice as to which area of the allocated slab to be de-allocated, the advice is returned to the program.

15. (Original) A system for squeezing slabs empty, a slab being a block of allocated memory space, the system comprising:

at least one storage device for storing code data; and

at least one processor for processing the code data to determine whether a slab is to be squeezed empty, and to preclude, if the slab is to be squeezed empty, data from being placed in any unused space of the slab.

16. (Original) The system of Claim 15 wherein data is precluded from being placed in any space in the slab that becomes unused anytime thereafter.

17. (Original) The system of Claim 16 wherein the slab is de-allocated when the slab becomes empty.

18. (Original) The system of Claim 17 wherein precluding data from being placed in an unused space of the slab includes disclaiming the unused space.

19. (Original) The system of Claim 18 wherein a collection of slabs is a pile, the pile having a maximum amount of allowable memory space that can be allocated thereto.

20. (Original) The system of Claim 19 wherein if an application reduces the maximum amount of allowable memory space of a pile and the current amount of memory space exceeds the reduced maximum, at least one of the slabs in the pile is targeted to be squeezed empty.

21. (Original) The system of Claim 20 wherein if the program requests advice as to which area of the allocated slab to be de-allocated, the advice is returned to the program.

(ix)

Evidence Appendix

No evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131 and 1.132 nor was there any evidence entered by the Examiner relied upon by Appellants in this appeal.

(x)

Related Proceedings Appendix

There are no decisions rendered by a court or the Board that would have a bearing on the Board's decision in the pending appeal.